VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Detekce význačných bodů obličeje za použití mobilních zařízení

Facial Landmarks Detection using Mobile Devices

2016 Michael Tichopád

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Michael Tichopád**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Detekce význačných bodů obličeje za použití mobilních zařízení
Facial Landmarks Detection using Mobile Devices

Jazyk vypracování: čeština

Zásady pro vypracování:

Detekce význačných obličejových bodů (tzv. landmarků) je v posledních letech stále více využívanou technikou. Z detekovaných zájmových bodů můžeme rozpoznávat například výraz ve tváři nebo míru natočení obličeje. Tyto body mohou také napomáhat při rozpoznávání obličeje. Cílem této práce je vytvoření aplikace na platformě Android, která bude detekovat základní důležité obličejové body.

1. Popište základní pojmy a metody v oblasti detekce obličejů a jejich důležitých bodů v obrazech.
2. Popište knihovnu OpenCV a popište její využití na platformě Android.
3. S pomocí knihovny OpenCV detektor obličejových bodů implementujte.
4. Experimentálně ověřte funkčnost, přesnost a rychlost řešení.
5. Své závěry řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

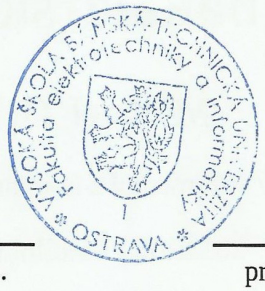
[1] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1, pp. I-511-I-518 vol.1 (2001)
[2] Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: ICB. pp. 828-837 (2007)
[3] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. vol. 1, pp. 886-893 vol. 1 (june 2005)
[4] M. Uricar, V. Franc and V. Hlavac, Detector of Facial Landmarks Learned by the Structured Output SVM, VISAPP '12: Proceedings of the 7th International Conference on Computer Vision Theory and Applications (2012)

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radovan Fusek**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016



doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Rád bych na tomto místě poděkoval panu Ing. Radovanu Fuskovi za cenné rady a vedení, svým blízkým za trpělivost a podporu a v neposlední řadě místní pražírně za jejich perfektní kávu. Bez zmíněných by tato práce nemohla vzniknout.

Abstrakt

Detekce význačných obličejových bodů (tzv. landmarků) je v posledních letech stále více využívanou technikou. Z detekovaných zájmových bodů můžeme rozpoznávat například výraz ve tváři nebo míru natočení obličeje. Tyto body mohou také napomáhat při rozpoznávání obličeje. Cílem této práce je vytvoření aplikace na platformě Android, která bude detekovat základní důležité obličejové body.

Klíčová slova: detekce, zájmové body, landmark, obličej, Android, bakalářská práce

Abstract

Facial points of interest (landmarks) detection has been increasingly used technique in last years. Obtained points of interest can be used, e.g., to analyze and distinguish facial expressions or rotation. These points can also help in face recognition. This work's purpose is to create an Android application capable od detecting basic facial points of interest.

Key Words: detection, points of interest, landmark, face, Android, bachelor thesis

Obsah

Seznam použitých zkratek a symbolů

LBP	–	Local Binary Patterns
BSD	–	Berkeley Software Distribution
MATLAB	–	Matrix Laboratory
PCA	–	Principal Component Analysis
SVM	–	Support Vector Machine
RGB	–	Red Green Blue
FPS	–	Frames Per Second
OCV	–	OpenCV
XML	–	eXtensive Markup Language

Seznam obrázků

Seznam tabulek

1 Úvod

Při stále rostoucí výkonnosti a snižující se velikosti výpočetních zařízení, se nabízí využití těchto zařízení pro implementaci technik analýzy obrazu. V případě této práce se tedy jedná o techniky detekce obličejů a význačných obličejových bodů. Tím rozumíme bodů, které definují obličej jako celek. V této práci jde především o základní body, přesněji: koutky úst, střed horního a spodního rtu, koutky očí a a středy horních a spodních víček, levý a pravý okraj jednotlivých obočí a špičku nosu. Jako základní je označuji proto, že jsou z mého pohledu nejpodstatnější pro jakoukoliv další práci se získanými daty.
Implementace praktického řešení je zaměřena na platformu Android a pro samotnou detekci bude tedy použita verze knihovny OpenCV pro tento systém.
První část této práce je věnována stručnému teoretickému rozboru obecné problematiky detekce obličejů, použité metody detekce obličejů v oblasti s nekontrolovaným pozadím a základní metody segmentace obrazu pomocí adaptivního prahování.
Druhá část se již soustředí na samotnou implementaci detektoru. Obsahuje rozbor problému, návrh postupu a samotný postup, který je tématicky rozdělen na dvě části detekce a to dle přístupu k řešení a použitých metod. V popisu řešení nejsou přítomny fragmenty zdrojového kódu, tento je dostupný v příloze na CD. Jsou zde však zmíněny metody, specifika knihovny OpenCV a zvláště při popisu inicializace této knihovny v prostředí Android je jim věnován prostor. V rámci tohoto prostoru je nutno zmínit OpenCV Manager - Android službu, která zajišťuje chod aplikací postavených nad touto knihovnou.
Součástí popisu implementace jsou postup a výsledky testování rychlosti detekce mobilní i desktopové verze aplikace a její přesnosti v porovnání s referenční databází.
V poslední části jsou zmíněny možnosti využití dat získaných touto aplikací v praxi, příklady existujících řešení a rozbor známých problémů této implementace a jejich možné řešení v budoucích pracech.

2 Metody v oblasti detekce obličejů a jejich význačných bodů v obrazech

2.1 Motivace

Detekce obličejů má dnes mnohá využití. Ať už jde o automatické přednastavení oblastí pro označení lidí na fotografiích, tak jako používají některé sociální sítě, samozaostřování fotoaparátů, nebo rozpoznávání emocí na základě analýzy jednotlivých význačných bodů obličeje. Pokud jsme schopni detekovat dostatečné množství detailních bodů, je možné využít tato data pro aplikaci augmentované reality, nebo dokonce rozpoznání jednotlivců dle jejich obličeje.
Z pozice zorniček je možné zjistit směr pohledu, což najde své využití například v automobilovém průmyslu, kdy může automobil sám zasáhnout do řízení při nepozornosti řidiče, nebo v marketingovém a designerském odvětví, pokud vememe v potaz možnost zjistit kde uživatel směřuje nejvíce svou pozornost. Na základě pozice špičky nosu a očí zase můžeme sledovat míru natočení obličeje a tak dále.
Knihovna OpenCV je pro implementaci řešení použita z důvodu své všestrannosti a popularity na poli počítačové analýzy obrazu. Je k dispozici pro většinu nejpoužívanějších platforem (Windows, Linux, Android, MacOS) s interface v jazycích C, C++, Python, Java a MATLAB. Navíc je dostupná pod licencí BSD, je tedy volně šířitelná.
Použití platformy Android se nabízí zejména z důvodu rozšířenosti tohoto systému. Tuto platformu dnes využívá celá škála mobilních (i desktopových) zařízení, a to od mobilních telefonů, tabletů a mini-pc (např. typu Raspberry PI), přes nositelnou elektroniku, domácí spotřebiče, bezpečnostní systémy až po automobilové systémy a tak dále.
Charakteristikou běžných mobilních zařízení stále bohužel je nižší výpočetní výkon, což je nutno zohlednit při řešení této práce.

2.2 Úvod k detekci obličejů

Detekce obličejů se zabývá přítomností obličejů v daném vstupním obraze (obvykle v odstínech šedi), a pokud jsou v tomto obraze přítomny, navrací jejich lokaci a obsah. Toto je první krok k další analýze informací obsažených v obličejích.
Algoritmy pro detekci obličejů se obvykle zaměřují na čelní pohled na obličeje. Při detekci na nekontrolovaném pozadí je základním postupem porovnávání oblastí vstupního obrazu s daty reprezentující lidský obličej v databázi. Tyto data jsou získány procesem rozpoznávání obličejů, který probíhá následujícím způsobem:

1. Detekce pozice obličejů
2. Normalizace obličejů[1]
3. Sběr charakteristik detekovaných obličejů
4. Předání těchto charakteristik algoritmu strojového učení

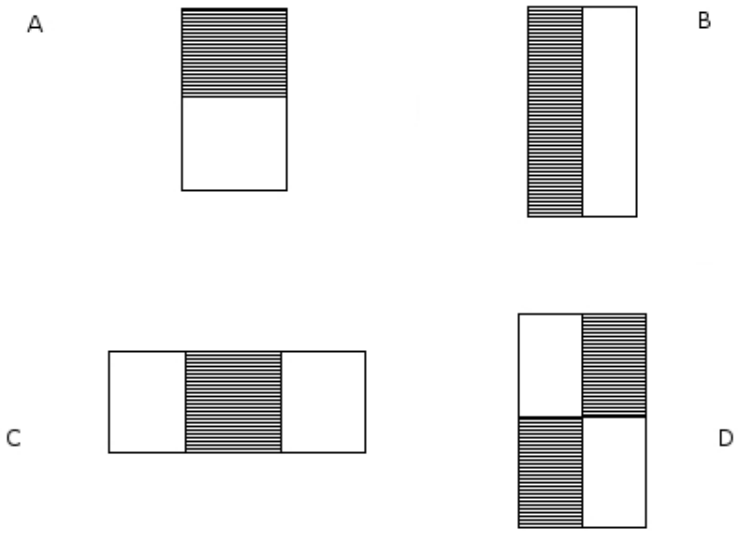
První krok je obvykle proveden klasickým Viola-Jonesovým algoritmem[3] pro objektovou detekci, který je rychlý a spolehlivý. Nalezené obličeje mohou mít odlišný jas, kontrast nebo velikost. Pro zjednodušení zpracování jsou všechny zmenšeny na stejnou velikost a jsou jim kompenzovány rozdíly v nasvícení (např. normalizací histogramu) v kroku druhém.
Ve třetím kroku existuje více možných metod. Dřívější detektory se pokoušely najít specifické pozice (středy očí, konec nosu, okraje rtů, atd.) a použít geometrické vzdálenosti a úhly mezi těmito pozicemi pro rozpoznávání (např. pomocí Hausdorffovy vzdálenosti [4, 5]). Tyto metody byly velmi rychlé, ne však natolik spolehlivé.
Aktuálnější metoda, tzv. *Eigenfaces*, je založena na tom, že obrazy obličejů mohou být aproximovány jako lineární kombinace základních obrazů (získaných skrze analýzu hlavních komponent[2] z velké sady tréninkových obrazů)[2]. Lineární faktory této aproximace mohou být použity jako charakteristiky. Tato metoda může být aplikována individuálně i na části obličeje (oči, nos, ústa). Nejlepších výsledků je takto dosaženo, pokud jsou pozice obličejů ve všech obrazech stejné. Pokud je u některých obličejů např. směr pohledy opačný, než u ostatních, výsledek nebude stejně kvalitní.
Čtvrtý krok je relativně přímočarý: Jsou k dispozici sady číselných bodů pro jednotlivé obličeje a obrazy obličejů získané tréninkem a je třeba najít tréninkový obličej, který je "nejpodobnější"vstupnímu obličeji. Zde nastupují algoritmy strojového učení, například support vector machine (SVM). Jako další můžeme zmínit umělé neuronové sítě nebo algoritmus k-nejbližších sousedů. Pokud jsou získané charakteristiky kvalitní, na volbě konkrétního algoritmu již příliš nezáleží.

[1] Normalizace v oblasti zpracování obrazu je proces snížení rozsahu hodnot intenzity pixelů. Příkladem použití jsou fotografie se slabým kontrastem kvůli oslnění. Normalizace je někdy označována jako roztažení kontrastu, nebo roztažení histogramu.[1]

[2] Analýza hlavních komponent (*Principal Component Analysis, PCA*) je v teorii signálu transformace sloužící k dekorelaci dat. Často se používá ke snížení dimenze dat s co nejmenší ztrátou informace.

2.3 Kaskádové klasifikátory

Momentálně nejpoužívanější metoda v obličejové detekci je metoda kaskádových klasifikátorů. Její základní implementace je přítomna v knihovně OpenCV a v rámci praktické aplikace výsledků této práce je použita pro detekci obličeje a jeho hranic. Algoritmus dvojice Viola a Jones používá kaskádů "slabých klasifikátorů", využívající jednoduchých Haarových příznaků a integrálního obrazu, která po tréninku algoritmem *AdaBoost*[3] dosahuje velmi dobrých výsledků. Protože metoda klasifikuje obrazy podle hodnoty jednoduchých příznaků, operuje mnohem rychleji, než kdyby pracovala přímo s pixely. Jsou zde použity tři druhy příznaků (viz Obrázek 1). Hodnota *dvou-obdelníkového příznaku* je rozdíl mezi součtem pixelů v obou obdelníkových oblastech. Oblasti maji stejnou velikost a jsou horizontálně či vertikálně sousední. *Tří-obdelníkový příznak* odečítá součet dvou krajních obdelníků od prostředního. A poslední, *čtyř-obdelníkový příznak* počítá rozdíl mezi diagonálními páry obdelníků.[3]

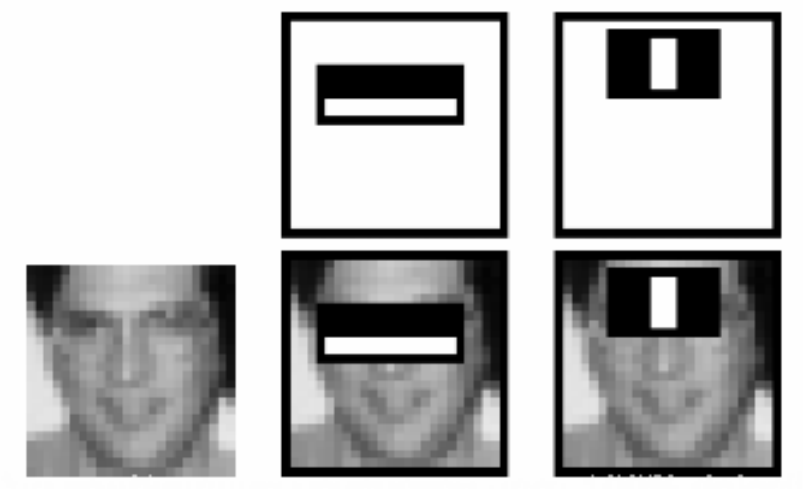


Obrázek 1: Příklady používaných Haarových příznaků.

Tyto příznaky jsou součástí *integrálního obrazu*, tedy jakéhosi výřezu, který se posouvá po vstupním obraze a vyhledává shodu (viz Obrázek 2). Uvedené druhy příznaků jsou použity z prostého důvodu. Bylo by velice neefektivní, pokud by měl algoritmus kontrolovat vstupní obraz na veškeré Haarovy příznaky (pro představu, vstup o velikosti 24x24 by ve výsledku dal přes 160 000 příznaků). Proto se algoritmus AdaBoost snaží vytvořit soustavu nejužitečnějších příznaků, které nejlépe klasifikují obličej a mají tedy nejmenší chybovost. Každý obraz je na začátku ohodnocen stejnou vahou.

[3]Jeden z nejpoužívanějších boostovacích algoritmů. Boostování je přístup k strojovému učení založený na principu tvorby vysoce přesných pravidel předpovědi kombinací množství relativně slabých a nepřesných pravidel.[9]

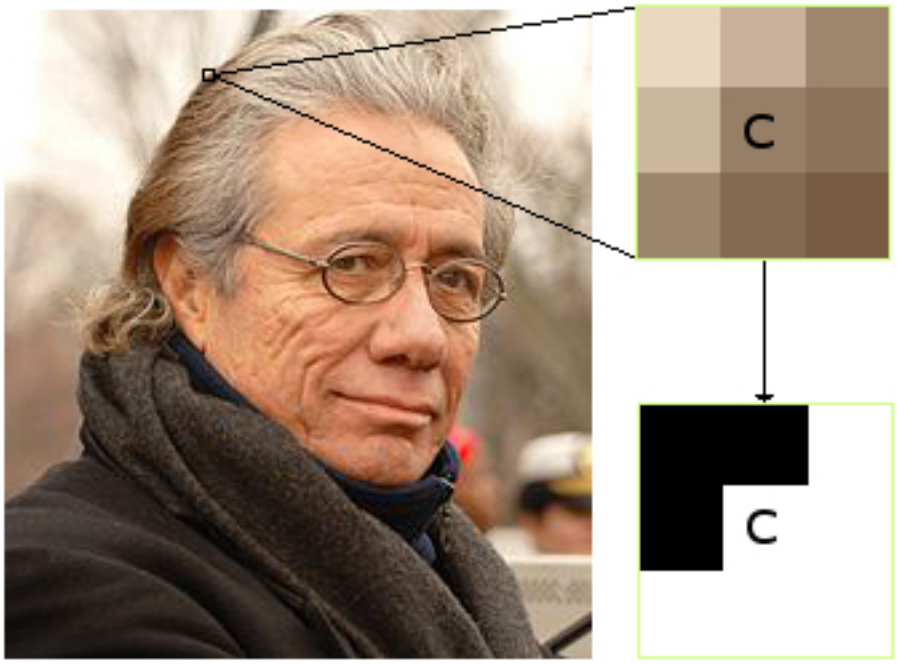
Po proběhnutí každé klasifikace je neklasifikovaným obrazům přidáno na váze. Tento proces se opakuje tolikrát, dokud není dosaženo požadované přesnosti či požadovaného množství příznaků.



Obrázek 2: Porovnávání vstupu s prvním a druhým příznakem vybraným alg. AdaBoost[3]

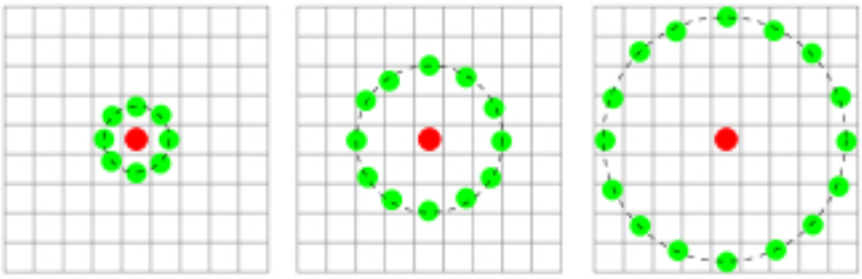
Konečný klasifikátor je vážený součet těchto lehkých (slabých) klasifikátorů. Tyto slabé klasifikátory jsou tak nazývány proto, že samy o sobě nedokáží klasifikovat obraz, ale spolu s ostatními tvoří silný klasifikátor.
Protože většinovou část vstupního obrazu tvoří ne-obličejové oblasti, je zapotřebí aplikovat metodu ke kontrole, zda-li je kontrolované okénko obličejovým regionem, nebo ne. Pokud by nešlo o obličejový region, oblast by se již dále nekontrolovala.
Z tohoto důvodu autoří zavedli koncept *kaskády klasifikátorů*. Místo aplikování všech dostupných klasifikátorů najednou, shrnují příznaky do jednotlivých stupňů (kaskád) a tyto aplikují postupně. Pokud okénko neprojde prvními stupni klasifikace, je vyřazeno a dále se již nepoužije. Počet příznaků v jednotlivých kaskádách se navíc stupňuje, což dále přispívá k urychlení algoritmu.[11]
Pro detekci pomocí kaskádových klasifikátorů je možné využít i data získaná tréninkem za pomocí příznaků LBP (lokální binární vzory). V tomto případě jsou jednotlivé příznaky reprezentovány histogramem. Při vytváření příznaku se prochází každý pixel (kromě krajních pixelů) a je porovnávána hodnota jeho intenzity s okolními pixely (viz Obrázek 3). Obvykle se jako vstup používají obrazy ve stupních šedi, ale je možné použít i barevné obrazy a porovnávat intenzity ve třech rozměrech.
Pokud je hodnota intenzity okolního pixelu nižší, než hodnota daného pixelu, je okolní pixel nastaven na 0 a naopak. Po porovnání všech okolních pixelů je celková binární hodnota této oblasti sečtena a následně je převedena na desítkové číslo. Jakmile je toto provedeno pro celý vstupní obraz, z histogramu jsou získány hodnoty unikátních příznaků.
Tyto charakterizují daný obraz. Příznaky LBP jsou odolné vůči změně osvětlení, protože poměry

hodnot mezi jednotlivými pixely a jejich okolím zůstanou neměnné. Díky tomu jsou také odolné proti změně rotace. Výhodou jejich použití pro kaskádovou klasifikaci oproti použití Haarových příznaků je bezpochyby větší rychlost samotné klasifikace. To je dáno faktem, že LBP příznaky provádějí své výpočty v celých číslech. Narozdíl od Haarových příznaků, které počítají s čísly desetinnými. Nevýhodou LBP pak může být nižší přesnost.



Obrázek 3: Příklad vytváření LBP příznaku. Procházený pixel je označen písmenem C.

Počet okolních pixelů ohodnocovací funkce může být i větší, než 8, jak je naznačeno na Obrázku 4 . Tímto se také zmenší výsledný počet vzorů.[8]

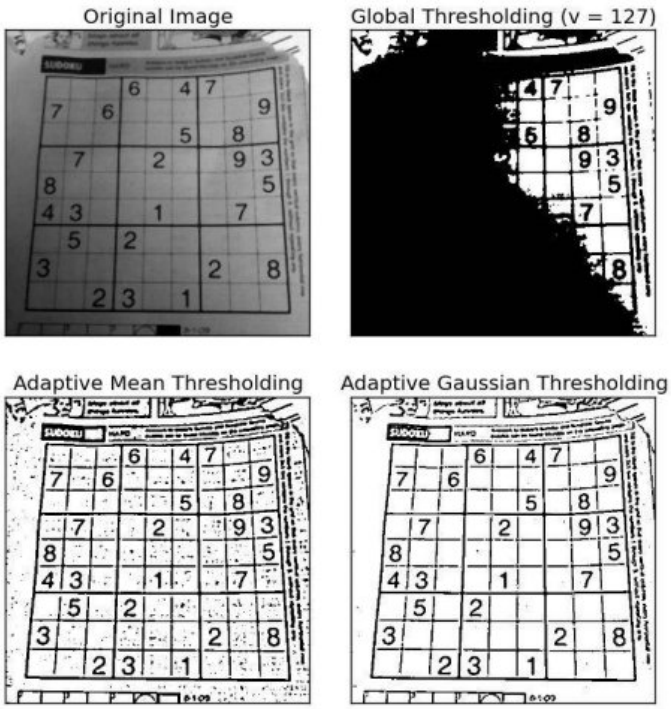


Obrázek 4: Tři příklady definice okolí pro spočítání LBP příznaku.

2.4 Adaptivní prahování

Jednoduché prahování v oblasti zpracování obrazu funguje na principu porovnávání hodnoty jednotlivých pixelů se stanovenou prahovou hodnotou. Pokud je hodnota pixelu nižší, než pra-

hová, je jeho hodnota nastavena na 0 (např. černá barva, pokud jde o obraz ve stupních šedi) a naopak, pokud je vyšší, je nastavena na 1 (např. bílá barva). Jednoduché prahování ovšem nebere v potaz jasové rozdíly ve zpracovávaném obraze, což se může ve výsledku projevit chybnou segmentací obrazu (viz Obrázek 5).
Zejména při segmentaci tak detailní oblasti, jakou je lidský obličej je proto mnohdy lepším postupem použití adaptivního prahování. V podstatě jde o stejný princip, jako u jednoduchého prahování. Rozdíl je v oblasti prahování a prahové hodnotě. Nedochází tady ke zpracování obrazu jako celku, ale obraz je rozdělen na podoblasti (bloky) o stanovené velikosti a tyto jsou prahovány zvlášť.



Obrázek 5: Rozdíly mezi metodami prahování.[12]

Prahová hodnota je dána dle dvou možných přístupů k adaptivnímu prahování, a to odečtením *konstanty C* buď od střední hodnoty pixelů v dané podoblasti nebo od váženého součtu sousedních hodnot, kde jsou váhy dány gaussovským okénkem[4]
Tímto jsou získána různá prahování pro různé oblasti a tak jsou lépe kompenzovány jasové rozdíly na vstupu.

[4]V oblasti zpracování signálu je funkce okénka matematická funkce, která je mimo vybraný interval ohodnocena nulou. Gaussovo okénko je dáno Fourierovou transformací Gaussovy funkce.

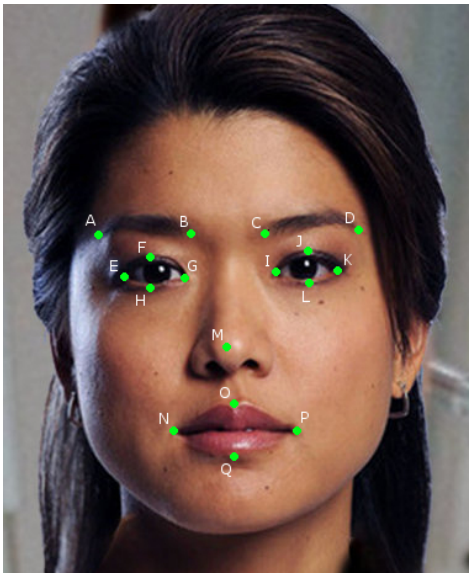
3 Implementace řešení a analýza meřených dat

3.1 Popis problému

Význačné body obličeje, jejichž detekce je cílem této práce jsou tedy:

- Koutky úst
- Střed horního a spodního rtu
- Koutky očí
- Střed horních a spodních víček
- Okraje obočí
- Špička nosu

K jejich získání je zapotřebí detekce hranic jednotlivých oblastí v obraze. Tyto oblasti budou zcela jistě ležet uvnitř obličeje a tedy prvním krokem bude detekce samotného obličeje. Pokud je toto splněno a podařilo se získat hranice oblastí obličejových rysů, dalším krokem je extrakce konkrétních rysů z každé jednotlivé oblasti. Posledním krokem tedy bude zisk význačných bodů z takto extrahovaných rysů. Na Obrázku 6 jsou vyznačeny zájmové body, které jsou cílem této detekce. Body A-D označují okraje obočí, body E-L koutky očí a středy očních víček, M špičku nosu a N-Q označují koutky úst a středy rtů.



Obrázek 6: Požadované zájmové body.

V potaz je nutno brát omezení mobilních zařízení, na které je tato implementace cílena, metody přístupné v použité knihovně OpenCV a v neposlední řadě platformu Android jako takovou.

Analýza obrazu a použité detekční algoritmy mohou být velmi náročné na výpočetní výkon, proto je potřeba najít způsob, jak dosáhnout co největší přesnosti detekce při co nejmenším vytížení procesoru.
Implementace bude řešena skrze knihovnu OpenCV verze 3.1.0 a jejího rozhraní v jazyce nativním pro prostředí Android, tedy v programovacím jazyce Java.
Samotná aplikace není cílena pro praktické nebo produkční nasazení, ale pouze pro experimentální účely. Tím pádem disponuje jen základním uživatelským rozhraním. Aplikace defaultně přebírá obraz ke zpracování z kamerového vstupu. Po zpracování těchto dat vykreslí detekované body do kamerového náhledu.
Zvolené vývojové prostředí je Android Studio verze 1.5.1 pro jeho efektivitu a kompatibilitu. Minimální podporovaná verze systému Android pro tuto aplikaci je 4.0.3 (Ice Cream Sandwich) a je použito pouze funkcí čistého Android API a knihovny OpenCV.

3.2 Příprava prostředí a inicializace OpenCV

Pro samotný vývoj OpenCV aplikací ve vývojovém prostředí Android Studio je nutné tuto knihovnu importovat do vytvořeného projektu. Soubory knihovny jsou importovány jako modul (viz dokumentace Android Studia[18]) a je potřeba nastavit samotné aplikaci závislost na modulu knihovny v nastavení struktury projektu.
Správu knihovny a chod aplikací nad ní vytvořených obstarává služba OpenCV Manager, která musí být přítomna na cílovém zařízení. Výzva k instalaci služby skrze Google Store proběhne automaticky při prvním spuštění aplikace využívající této knihovny.
Pro práci s kamerovým vstupem je nutné, aby třída dané aktivity implementovala rozhraní *CvCameraViewListener2*[21]. Toto rozhraní deklaruje metody potřebné, jak ke zpracování obrazu z kamerového vstupu a jeho zobrazení, tak loader callback[5] metody k načtení služby OpenCV Manager, kontrole verze knihovny a zpracování souvisejících chyb.
Následujícím krokem je načtení souborů klasifikačních dat, které budou použity pro detekci kaskádovými klasifikátory. Tyto data jsou k dispozici jako součást knihovny OCV a to většinou jako data získaná tréninkem pomocí Haarových příznaků, ale použity jsou i data ziskaná tréninkem pomocí LBP příznaků, jenž jsou součástí starší verze knihovny (ver. 2.4.11).
Soubory dat jsou k dispozici ve formátu XML, pro který Android API v době řešení této práce nenabízí přímé metody načtení souboru jako celku. Je tedy nutné vytvořit metodu, která bude XML načítat pomocí třídy *FileOutputStream*[19] jako proud bytů a zapíše do dočasného souboru.
Teprve poté, co úspěšně proběhne inicializace služby OpenCV Manager, je možno inicializovat veškeré specifické datové typy knihovny, jako např. objekty typu *CascadeClassifier*[22] reprezentující OCV implementaci kaskádových klasifikátorů. Při inicializaci objektů typu CascadeClassifier jim jsou jako argument předány dočasné soubory klasifikačních dat.

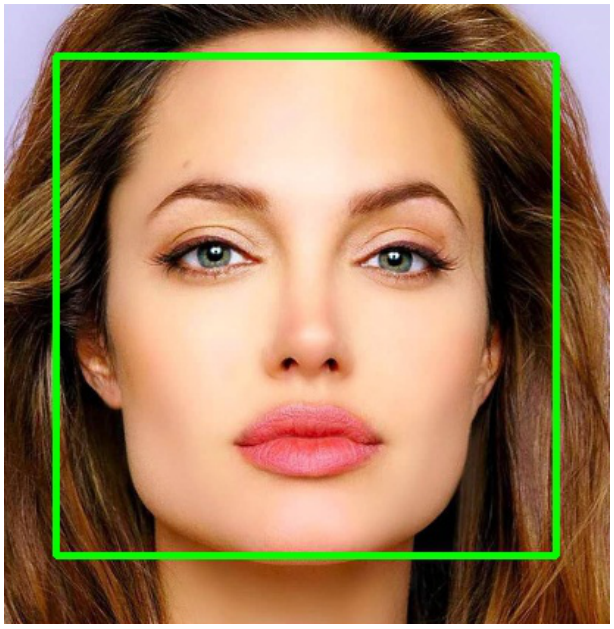
[5] Callback metody jsou předávané jako argument jiným metodám, které tento argument volají za daných podmínek.

3.3 Obrazový vstup

Získání obrazu z kamerového vstupu a jeho zobrazení probíhá skrze třídu *JavaCameraView*[20], která také umožňuje volbu kamerového zařízení a zobrazení počtu snímku za sekundu (FPS). Zpracování obrazového vstupu přes tuto třídu probíhá přes výše uvedené rozhraní. Nejdůležitější metoda pro zpracování obrazu deklarovaná tímto rozhraním je metoda *onFrame()*[21], která je volána při každém úspěšně získaném vstupu z kamerového zařízení. Získaný obraz je ve formátu datového typu *Mat*[23]. V této metodě je řešena naprostá většina implementace. Je důležité zmínit, že OCV třída pro zobrazení kamerového vstupu obsahuje v době tvorby této práce nahlášenou chybu, která některým zařízením neumožňuje zobrazení při orientaci displeje na výšku. Z toho důvodu je v aplikaci nastavena defaultní orientace na šířku.

3.4 Detekce obličeje a jeho hranic

Jak již bylo zmíněno, k detekci obličeje je použita OCV implementace kaskádové klasifikace - funkce *detectMultiScale()*[22]. Hodnoty argumentů funkce jsou optimalizovány s ohledem na výkon mobilního zařízení. Proto jsem se rozhodl nedetekovat více, než jeden obličej najednou. Tohoto je dosaženo předáním argumentů *minNeighbors* a *minSize* funkci, specifikujících minimální počet sousedních detekcí, pro zachování jednotlivé detekce (vyšší hodnotou je dosaženo menšího počtu kvalitnějších detekcí) a minimální velikost detekovaného objektuvzhledem k velikosti vstupního obrazu. Minimální velikost obličeje je počítána jako 50% výšky vstupního obrazu.
Pro detekci obličeje jsou použity data ziskaná tréninkem LBP příznaků a to z důvodu výrazně rychlejší detekce na mobilním zařízení (více v kap. 2.3).

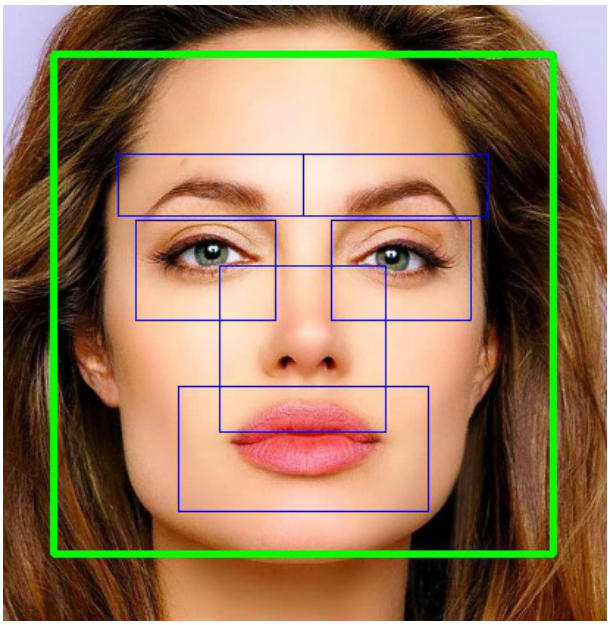


Obrázek 7: Vyznačené hranice detekovaného obličeje.

Nicméně je takto detekce o něco méně přesná, což se projevuje náhodnou změnou velikosti hranic detekovaného obličeje.
Výsledkem detekce je matice obdélníků, kdy každou úspěšnou detekci zastupuje právě jeden obdelník. Další práce s oblastí obličeje je tedy prováděna průchodem této matice v cyklu a zpracováním jednotlivých detekcí.

3.5 Segmentace zájmových oblastí

Jednotlivé zájmové oblasti, ze kterých budou extrahovány obličejové rysy, jsou segmentovány staticky jako podmatice obličeje. Jejich velikost a lokace je dána čistě jako zlomek rozměrů obličejové oblasti (viz. Obrázek 8).
Poměry užité pro tyto výpočty jsou založeny na antropometrických hodnotách rozdělení obličeje[6] a dále upraveny, aby vznikly rezervy pro pohyb obličejových rysů při detekci v reálném čase.
Takto řešená segmentace je nesmírně výhodná oproti, např. detekci oblastí rysů pomocí kaskádových klasifikátorů, co se týče požadavků na výpočetní výkon, o což jde u mobilních zařízení především.
Tento přístup však má i významné nevýhody. Hlavní nevýhoda pramení ze samotného charakteru *statického* rozdělení, tedy takto rozdělené oblasti vystihují poměry určitého průměrného obličeje, ale absencí flexibility mohou u některých případů vykazovat nepřesnost. Také může dojít k chybým detekcím při výraznějších rotacích obličeje a pro kvalitní detekci je zapotřebí čelního pohledu. I přes to jsem se, vzhledem k výkonu aplikace, rozhodl použít toto řešení.



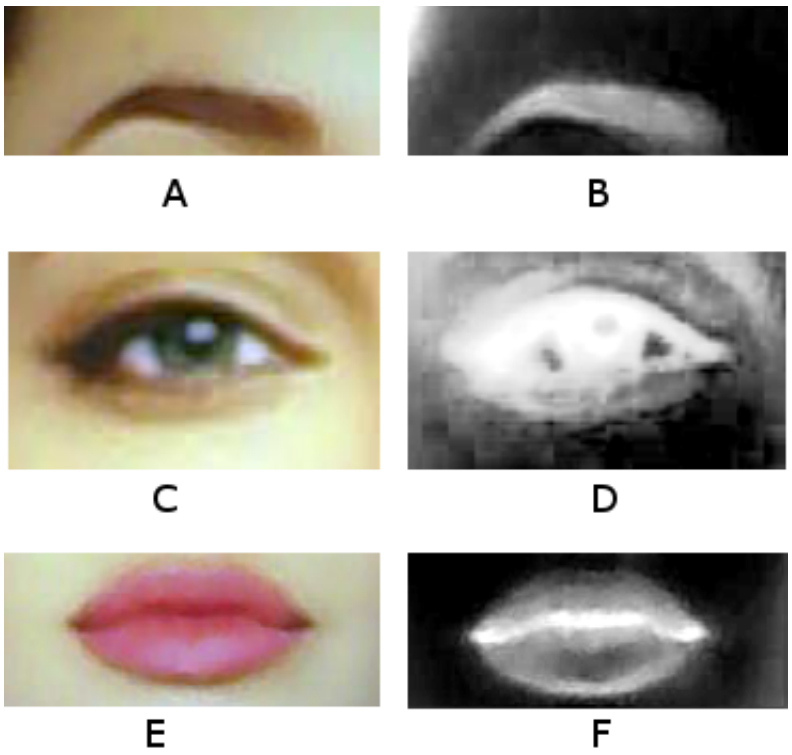
Obrázek 8: Obličej rozdělený na oblasti zájmu.

3.6 Detekce zájmových bodů očí, úst a obočí

Detekce zájmových bodů je realizována technikami segmentace obrazu podle barev. Tyto obličejové rysy je možné odlišit od okolních oblasti na základě rozdílů v barevném spektru, ale stejný postup není možné aplikovat pro detekci špičky nosu. Je sice možné takto detekovat nosní dírky, ale ty nejsou vždy viditelné. Proto je následující postup rozdělen na detekci ostatních rysů a detekci špičky nosu, která je řešena jiným způsobem (viz. kapitola 3.7).

3.6.1 Výběr barevného kanálu a normalizace histogramu

Je známo z literatury[10], že pixely reprezentující lidskou kůži mají vysokou intenzitu červené barvy v RGB barevném spektru. Proto, pro dosažení co největšího odstupu oblastí nereprezentujících kůži od okolí, je oblast rysu rozdělena na jednotlivé barevné kanály (modrý, zelený a červený kanál) a k dalšímu zpracování je již použito pouze invertovaného červeného kanálu.
Oblast rtů je zpracovávána převedením na odstíny šedé, protože pixely reprezentující rty mají samy o sobě vysokou intenzitu červené barvy.
Jasové rozdíly v jednotlivých oblastech by mohly zapříčinit chybné detekce, proto je nutné rozdíly kompenzovat normalizací histogramu (viz. poznámka v kap.2.2). Na Obrázku 9 jsou znázorněny jednotlivé zájmové oblasti v invertovaném červeném spektru s normalizovaným histogramem:



Obrázek 9: Normalizované oblasti v invertovaném červeném spektru.

3.6.2 Rozostření a prahování

Aby bylo možné jasně stanovit okraje rysů a jejich okolí, jsou zájmové oblasti segmentovány pomocí adaptivního prahování (viz. kapitola 2.4). Velikost bloku a konstanty C pro adaptivní prahování byla stanovena experimentálně tak, aby bylo dosaženo ideálního tvaru blobu[6], reprezentujícího daný rys, pokud možno u co největšího počtu testovaných obličejů. Takto získaný blob bude sloužit k dalšímu zpracování (viz. Obrázek 10).

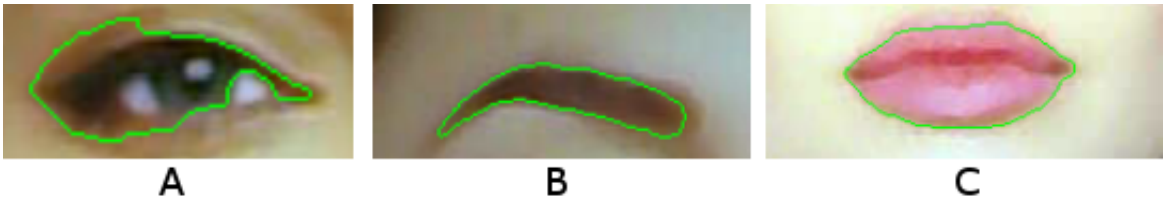
Obrázek 10: Bloby získané rozostřením a následným prahováním.

Vzhledem k tomu, že defaultně je zpracováván obraz z kamerového zařízení, hranice blobu budou s každým novým snímkem neustále kmitat v rámci jednotek pixelů. Pro stabilizaci hranic blobu je použito lehkého gaussova rozostření ještě před samotným prahováním. Tento krok je důležitý pro další fáze detekce.

3.6.3 Zisk největší kontury

Z binárních obrazů jednotlivých oblastí, které jsou k dispozici, je možné detekovat kontury. Kontury jsou body ohraničující každý samostatný blob v obraze. V knihovně OpenCV je implementován Suzukiho algoritmus pro jejich detekci[7] v metodě *findContours()* třídy *Imgproc*. Tato implementace detekuje bílé bloby na černém pozadí a vrací seznam matic bodů (x, y), kde každá matice bodů představuje právě jednu konturu ohraničující právě jeden blob.
Je pravděpodobné, že ve zpracovávané zájmové oblasti bude detekováno více, než jedna kontura. Kromě kontury detekovaného rysu mohou být detekovány menší bloby, šum a různé okrajové zásahy. Předpokládá se, že právě detekovaný rys zabírá největší plochu a tedy jeho kontura bude zabírat rovněž největší plochu. Dalším krokem je tedy zisk největší kontury.
Průchodem seznamu získaného detekcí je možné zjistit velikost plochy jednotlivých kontur pomocí metody *area()*. Porovnáváním velikosti ploch kontur lze získat matici bodů představující největší konturu a tedy body ohraničující detekovaný rys. Takto získané největší kontury jsou naznačeny na Obrázku 11. Mezi těmito body jsou obsaženy cílené zájmové body.

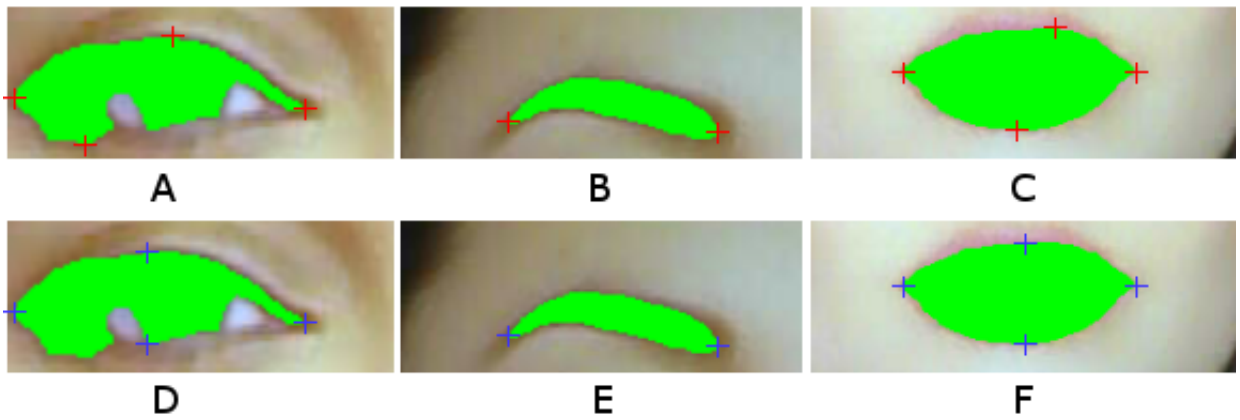
[6] Bloby v oblasti zpracování obrazu označují jednolité shluky pixelů kladných hodnot v binárním obraze.



Obrázek 11: Vykreslení největších detekovaných kontur.

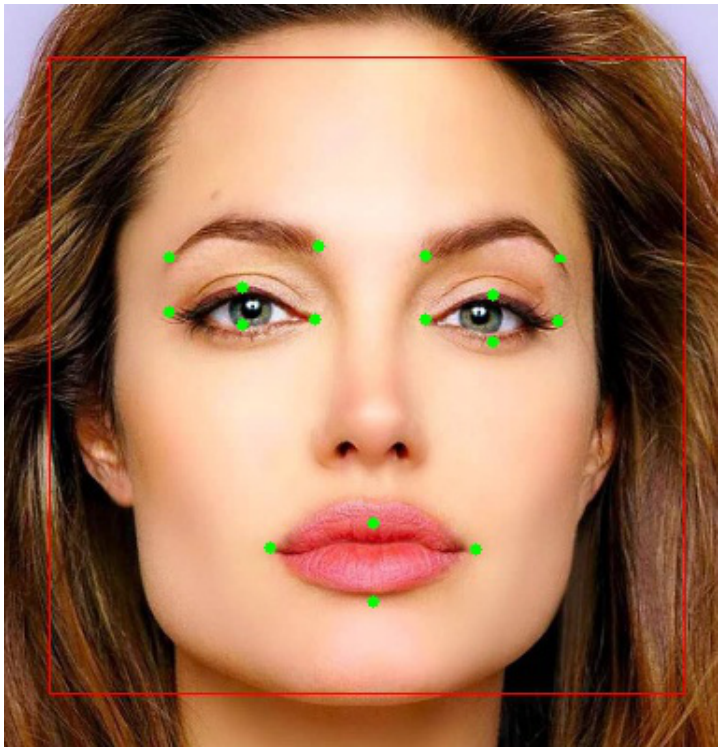
3.6.4 Zisk zájmových bodů

Podle stanovení zájmových bodů z podkapitoly 3.1 je možné, při pohledu na tvary blobů, logicky odvodit pozice těchto bodů v matici reprezentující tyto bloby. Půjde o tzv. extrémní body, tedy body, jejichž hodnoty x a y jsou na osách hodnot krajní. Získat je lze průchodem matice kontury a porovnáváním jednotlivých bodů. Protože matici bodů nelze standartně procházet, nabízí OCV třída *MatOfPoint* metodu *toArray()* pro převedení matice na pole.
Pro hodnoty x je podmínka prostá - musí jít o extrémní krajní body. Pro hodnoty y jít o extrémní krajní body nemůže, protože vzhledem k tvaru oka se tyto body budou neustále rozcházet a v případě analýzy na základě těchto bodů by, např. nebylo možno s jistotou stanovit pozici očních víček. Proto musí u y-ových hodnot jít o krajní body v polovině oka. Rozdíl mezi těmito přístupy je ručně znázorněn na Obrázku 12.
Aby bylo možné zjistit pozici "poloviny oka", je nejdříve potřeba stanovit rozměry oka. Toho je dosaženo použitím statické metody *boundingBox()* OCV třídy *Imgproc*.
Tato metoda vrací obdelník opsaný dané kontuře. Rozměry tohoto obdelníku tedy odpovídají maximální šířce a výšce kontury a pozice "poloviny oka"se potom rovná pozici poloviny obdelníku.
U zájmových bodů očí a úst jsou tedy hodnoty x získány jako krajní a hodnoty y jako krajní v x-ové pozici poloviny obdelníku prohledávané kontuře opsaného.



Obrázek 12: Rozdíl mezi extrémními body (A, B, C) a zájmovými body (D, E, F).

Získané zájmové body jsou vykresleny do vstupního obrazu pomocí statické metody *circle()* třídy *Imgproc*, tyto body jsou znázorněny na Obrázku 13 níže.



Obrázek 13: Detekované zájmové body očí, úst a obočí.

3.7 Detekce špičky nosu

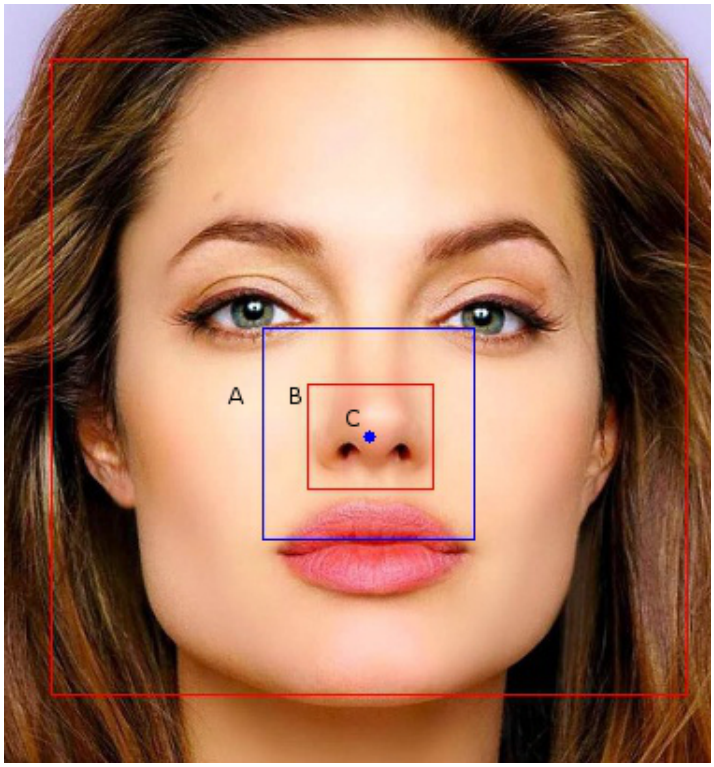
Jak již bylo poznamenáno v podkapitole 3.6, u nosu není aplikován stejný postup detekce zájmových bodů, jako u ostatních rysů. Proto i přes výpočetní náročnost kaskádové detekce, je v tomto případě použita.

Pro detekci jsou použity Haarovy příznaky a to ze dvou důvodů: součástí OpenCV knihovny nejsou, v době tvorby této práce, data pro detekci nosu z tréninku jiných příznaků, než Haarových a vzhledem k tomu, že tyto data detekují jen nos obecně, ale ne konkrétně jeho špičku, je zde zapotřebí větší přesnosti, resp. lepší stability získaných hranic. Špička nosu bude totiž stanovena jako pevný bod uvnitř těchto hranic.

Z hlediska výkonnosti je nutné učinit určitou optimalizaci, a to zúžením prohledávané oblasti ve vstupním obraze a podobně jako u detekce obličeje, snížením počtů detekcí a výpočtem minimální velikosti detekovaného objektu. Minimální velikost je počítána jako 20% z výšky hranic obličeje. Přesto však má takto implementovaná detekce nosu významný vliv na snížení rychlosti detekce zájm. bodů pro celý obličej.

Zúžená oblast je mimo jiné důležitá proto, že eliminuje jakékoliv chybné detekce, které by se mohly projevit mimo přirozenou oblast nosu.

Oblast definovaná maticí obdelníků, která je výstupem OCV funkce pro kaskádovou detekci, je na takové pozici, kde se špička nosu vyskytuje zhruba ve středu oblasti. Zájmový bod je tedy získán jako střed takto detekovaného ohraničení.
Na Obrázku 14 je vykreslena oblast zúžená pro detekci (A) a oblast získaná kaskádovou detekcí (B). Zájmový bod bude poté vykreslen do středu detekovaných hranic.



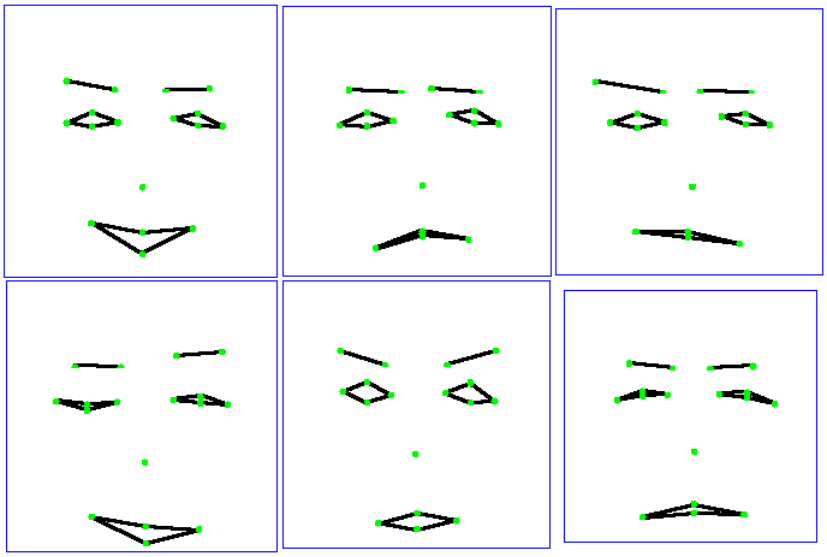
Obrázek 14: Detekce špičky nosu.

3.8 Příklady výsledné detekce

Na Obrázku 15 jsou bílou barvou naznačeny hranice detekovaného obličeje a zelenou barvou vyznačeny zájmové body detekované aplikací.

Obrázek 15: Ukázka funkční detekce.

Níže jsou tyto body abstrahovány od obličeje a spojeny v celky pro názornost. Z Obrázku 16 je znatelné, jak mohou být tyto zájmové body použitelné, např. při rozpoznávání emocí.



Obrázek 16: Detekované body abstrahované od obličeje.

3.9 Měření přesnosti a rychlosti detekce

3.9.1 Přesnost detekce

Měření přesnosti detekce zájmových bodů je provedeno na základě obličejové databáze společnosti *BioID* [13]. Tato databáze je volně použitelná pro výzkum a testování a je součástí projektu *FGnet* (Face and Gesture Recognition Working Group).
Součástí databáze je sada velkého množství obrazů zachycující obličeje a sada textových souborů obsahující souřadnice ručně zaznačených ideálních zájmových bodů. Obrazy jsou ve stupních šedi o rozlišení 384x286 pixelů. Protože tato implementace používá k přesnější detekci červeného kanálu barevného spektra, vstup ve stupních šedi může mít vliv na výslednou přesnost.
Vyhodnocení přesnosti je provedeno na základě porovnání těchto ideálních bodů a bodů získaných mou detekcí. Aby bylo možné tyto body mezi sebou porovnat, byla aplikace upravena tak, aby načítala veškeré obrazy ze složky a následně pro každou úspěšnou detekci vytvořila soubor s názvem obrazu a do něj vypsala jednotlivé souřadnice detekovaných bodů.
Je nutné zmínit, že BioID databáze neobsahuje naznačené zájmové body středů očních víček, tím pádem ani jejich souřadnice. Tyto body jsou tedy při porovnávání vynechány.
Dalším krokem byl průchod veškerých vygenerovaných souborů a porovnání souřadnic se souřadnicemi z databáze. Výsledky jsou uvedeny v tabulkách níže.

Aby došlo k pokusu o detekci zájmových bodů, musí dojít nejprve k detekci obličeje. Detekce zájmového bodu je úspěšná, pokud náleží souřadnice bodu intervalu ideálních souřadnic z databáze s maximální odchylkou +- 10 pixelů.
Nejdříve je testována implementace používající kask. detekci obličeje za pomoci dat získaných LBP příznaky. Použita je testovací sada 409 obrazů z databáze. Pro každý obličej je detekováno 13 zájm. bodů.

Typ detekce	Úspěšné pokusy	Celkem pokusů	Podíl
Obličej	391	409	95,6%
Zájmové body	3893	4967	78,38%

Tabulka 1: Celková přesnost detekce při použití LBP příznaků.

Druhá je testována implementace používající Haarových příznaků pro detekci obličeje. Použita je testovací sada taktéž 409 obrazů z databáze.

Typ detekce	Úspěšné pokusy	Celkem pokusů	Podíl
Obličej	404	409	98,78%
Zájmové body	4007	5155	77,73%

Tabulka 2: Celková přesnost detekce při použití Haarových příznaků příznaků.

Z Tabulek 1 a 2 je možné si všimnout, že z 391 provedených úspěšných detekcí obličeje pomocí

LBP bylo provedeno 4967 detekcí jednotlivých bodů. Pro každý obličej ovšem proběhne 13 detekcí zájm. bodů, což dává celkem 5083 možných detekcí.
Fakt, že tedy zhruba 2,2% detekcí chybí, může být způsobeno chybnými případy detekce. U detekce pomocí Haar. příznaků by mělo být provedeno 5252 detekcí bodů oproti 5155 provedených, což dává zhruba 1,8% chybných detekcí.

Tato statistika rozdělena na jednotlivé případy, viz. Tabulka 3 a Tabulka 4. Případy jsou shrnuty na detekci jednotlivých očí a obočí, úst a nosu, místo výpisu jednotlivých zájmových bodů. Při stanovování úspěšnosti je každý bod ohodnocen jako část celku daného rysu. Například, úspěšná detekce levého koutku levého oka přidá hodnotu 0,5 k celkovým úspěšných detekcím bodů levého oka a v každém případě přidá tuto hodnotu k celkovému počtu pokusů.

Detekce zájm. bodů	Úspěšné pokusy	Celkem pokusů	Podíl
Levé obočí	341	389	87,66%
Pravé obočí	323	389	82,9%
Levé oko	296	389	76,1%
Pravé oko	311	389	79,95%
Špička nosu	298	389	76,6%
Ústa	264	367	71,9%

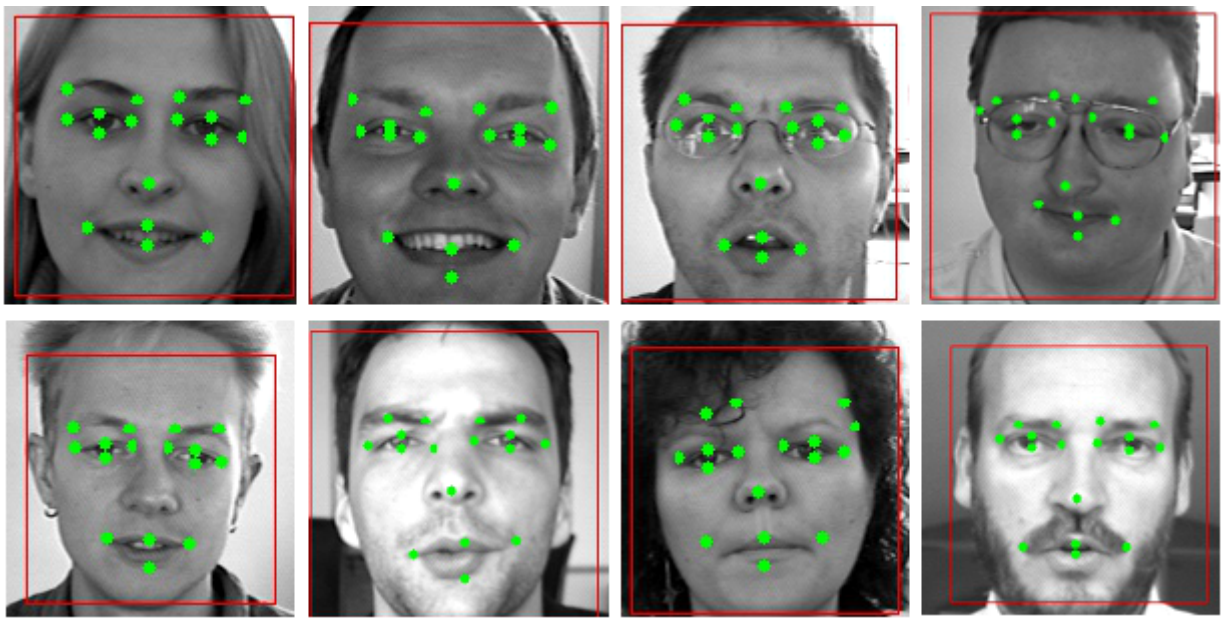
Tabulka 3: Přesnost detekce zájm. bodů při det. obličeje pomocí LBP.

Detekce zájm. bodů	Úspěšné pokusy	Celkem pokusů	Podíl
Levé obočí	358	405	88,4%
Pravé obočí	333	405	82,1%
Levé oko	323	405	79,75%
Pravé oko	319	405	78,77%
Špička nosu	295	405	72,84%
Ústa	262	378	69,34%

Tabulka 4: Přesnost detekce zájm. bodů při det. obličeje pomocí Haarových příznaků.

Z rozdílů mezi těmito dvěma přístupy je patrné, že při použití Haarových přízn. dosáhneme přesnější detekce hranic obličeje. Výsledná přesnost zájmových bodů je sice nepatrně menší, ale v důsledku přesnější detekce obličeje k ní bude docházet častěji a rozdíly mezi těmito přístupy se smažou.

Faktorem také může být sada dat získaných těmito příznaky, jež je dostupná v samotné knihovně OCV. Je možné, že vlastním tréninkem jakýchkoliv z těchto dvou typů příznaků by šlo dosáhnout lepších výsledků. Pokud je tedy rozdíl přesnosti těchto dvou přístupů zanedbatelný, důležitý je rozdíl v rychlosti. Na Obrázku 17 jsou ukázány příklady úspěšných i neúspěšných detekcí na obrazech z databáze. Jak je zde vidět, zejména při zpracování černobílého obrazu má implementace problém s významnými zásahy do zájmových oblastí (vlasy přes obočí, výrazné vousy, atp.).



Obrázek 17: Příklady detekcí nad obrazy z databáze BioID.

3.9.2 Rychlost detekce

Snížení náročnosti na výpočetní výkon je pro tuto mobilní aplikaci prioritní. Zejména pokud jde o kamerový vstup v reálném čase. Při zpracování obrazu ale velká část výkonu spočívá právě na rychlosti kamery. Naprostá většina zbytku výkonu je poté již na straně procesoru.
Výsledky viz. Tabulky 5 až 7. Pro porovnání je testována i desktopová verze aplikace (Tab. 7). Testovány jsou stavy s vypnutou detekcí, kompletní detekcí atd. Pokud není uvedeno jinak, při dalších stavech jsou aktivní pouze detekce uvedených rysů. Pro porovnání vlivu použitých typů příznaků na rychlost detekce je testována i implementace s Haarovými příznaky (Tab. 8 a Tab. 9).
Z Tab. 6 je znatelné, kdy výkon aplikace brzdí rychlost kamerového zařízení. Při kompletní a žádné detekci je propad výkonu pouze poloviční, narozdíl od Tab. 5, kde je propad prakticky čtyřnásobný.

Desktopová verze aplikace je algoritmicky totožná s mobilní. Vytvořená je v jazyce Java a používá stejnou verzi OpenCV knihovny (ver. 3.1.0).

Testované zařízení	**ST-Ericsson U8500 (2x Cortex-A9 1GHz)** 512 MB RAM Rozlišení kam. vstupu 480x320
Druh detekce	Průměrný počet snímků za sekundu
Žádná	**27,3**
Kompletní	**5,6**
Kompl. bez nosu	17,7
Obličej	23,9
Oči	19,3
Obočí	20,1
Ústa	19,8
Nos	6,2

Tabulka 5: Výkon aplikace na mobilním zař. s procesorem ST-Ericsson U8500.

Testované zařízení	**QC Snapdragon 210 (4x Cortex-A7, 1.1 GHz)** 1GB RAM Rozlišení kam. vstupu 800x480
Druh detekce	Průměrný počet snímků za sekundu
Žádná	**11,3**
Kompletní	**5,5**
Kompl. bez nosu	9,9
Obličej	10,1
Oči	9,9
Obočí	10,1
Ústa	9,8
Nos	6,4

Tabulka 6: Výkon aplikace na mobilním zař. s procesorem Snapdragon 210.

Výkon při použití Haarových příznaků pro detekci obličeje je testován pouze na jednom mobilním zařízení a jednom desktopovém zařízení, pro názornost rozdílu rychlosti zpracování operací nad desetinnou čárkou, které Haarovy příznaky používají (Tabulky 8 a 9 níže).
U mobilního zařízení není rozdíl mezi rychlostmi kompletní detekce tak významný. Nicméně rozdíl mezi kompletními detekcemi (bez detekce špičky nosu), při použití těchto dvou druhů příznaků, je zhruba dvojnásobný. Tento faktor jen potvrzuje důvod, proč jsou v této práci použity LBP příznaky pro detekci obličeje.
Rychlost detekce desktopové verze aplikace již není tolik ovlivněna použitými příznaky, ačkoliv je možné, že i zde by se projevily výkonnostní rozdíly při rozsáhlejších detekcích. Přesto z výsledků vyplývá horší schopnost mobilních zařízení operovat s desetinnými čisly.

Testované zařízení	**AMD C-50 (2x 1.0 GHz)** 2GB RAM Rozlišení kam. vstupu 640x480
Druh detekce	Průměrný počet snímků za sekundu
Žádná	**33,1**
Kompletní	**10,7**
Kompl. bez nosu	22,8
Obličej	31,2
Oči	25,8
Obočí	26,4
Ústa	24,5
Nos	14,5

Tabulka 7: Výkon desktopové verze aplikace na procesoru AMD C-50.

Testované zařízení	**ST-Ericsson U8500** Použity **Haarovy** příznaky
Druh detekce	Průměrný počet snímků za sekundu
Žádná	**26,7**
Kompletní	**3,2**
Kompl. bez nosu	8,1
Obličej	11,2
Oči	10,3
Obočí	10,5
Ústa	11,3
Nos	4,1

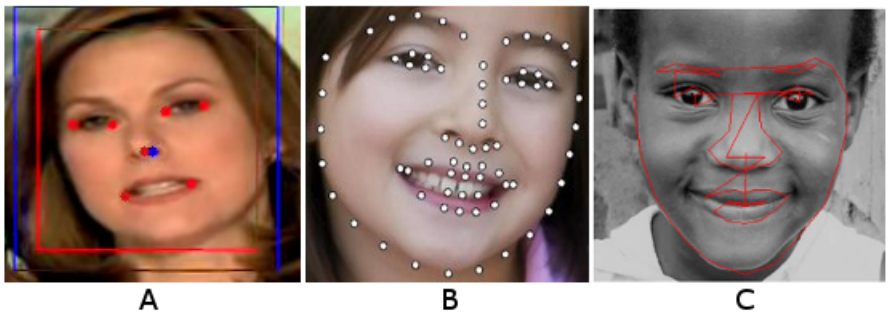
Tabulka 8: Výkon aplikace na mobilním zařízení při použití Haar. přízn.

Testované zařízení	**AMD C-50 (2x 1.0 GHz)** Použity **Haarovy** příznaky
Druh detekce	Průměrný počet snímků za sekundu
Žádná	**32,4**
Kompletní	**9,8**
Kompl. bez nosu	21,5
Obličej	30,62
Oči	23,2
Obočí	23,16
Ústa	23,41
Nos	11,7

Tabulka 9: Výkon desktopové verze aplikace při použití Haar. příznaků

4 Možné využití v reálných aplikacích a budoucí práce

Jak již bylo naznačeno v kapitole 2.1, detekce obličejů má již dnes velkou řadu využití. Detailní detekce obličejových bodů v podstatě odpovídá detailní detekci obličejů jako takových. A pokud dokážeme detailně strojově detekovat a analyzovat lidský obličej v reálném čase, je možné tyto informace využít, jak ke zlepšení interakce člověka a stroje, k identifikaci a bezpečnostním aplikacím, tak kupříkladu k jednoduššímu mapování 3D obrazu na lidský model.
V této době již existují hotové či stále vyvíjené implementace detekce význačných bodů, např.: implementace *flandmark* [14] dvojice Uřičář a Franc z ČVUT detekující špičku nosu, koutky úst a očí v reálném čase. Momentálně nahrazen jejich open-source projektem *CLandmark* [15] schopným detekovat větší množství zájmových bodů. Také implementace za pomocí hlubokého multi-taskového učení [16], od kolektivu Čínské univerzity v Hong Kongu, která dokáže velmi detailně detekovat až 68 obličejových bodů, nebo C++ knihovna *Stasm* [17] postavená na knihovně OpenCV a další.
Na Obrázku 18 jsou příklady výsledků detekce pomocí některých výše uvedených implementací: flandmark (A), implementace univerzity v Hong Kongu (B) a Stasm (C).
Tyto projekty používají mnohdy velmi odlišného řešení, aby dosáhly požadovaného výsledku. Nicméně žádná z těchto větších, či pokročilejších implementací, buďto nezaměřuje primárně mobilní zařízení a jejich výkon při zpracovávání v reálném čase, nebo s těmito zařízeními nepočítá vůbec. Přesto je právě toto odvětví stále více aktuální a má potenciál pro podobné typy aplikací.



Obrázek 18: Ukázka detekce existujícími implementacemi.

Tato práce se potýká s pár problémy, které vyplývají z její charakteristiky, přesto nejsou neřešitelné. Největším problémem je statické rozdělení obličeje3.5, které nemůže ze své podstaty přesně odpovídat rozložení všech obličejů. Při minimálním snížení výkonnosti by se teoreticky dalo statické rozlišení zaměnit kaskádovou detekcí pomocí LBP příznaků. Tyto příznaky nejsou prozatím přístupné jako součást OpenCV, ale je možné vytrénovat vlastní sadu. Tento přístup by mohl pomoci s problémem omezení rotace obličeje, který rovněž vyplývá ze statického rozdělení.
Další problém spočívá v zásazích výrazných předmětů, vlasů a vousů do zájmových oblastí, kdy

není v jednom zpracovávaném spektru možné rozlišit tyto zásahy od zájmových bodů, které s nimi splývají. Řešení tohoto problému by mohlo přínést zpracovávání více barevných kanálů najednou, nebo vyhodnocování pozice blobu vůči centru dané oblasti, což ovšem má opět vliv na rychlost detekce. To vše musí být, pro kvalitní výsledky, při další práci vycházející z této implementace, zváženo.

Přesto, že zmiňuji v kap. 2.1 detekci a analýzu pozice zorniček, v této práci není implementována. I tak je možné stejným postupem, jaký je použit pro detekci očí, s relativní přesností zjistit pozice zorniček pouhou změnou prahovacích hodnot.

U zařízení podporujících výpočty skrze grafické jádro navíc knihovna OpenCV nabízí možnost akcelerovat zpracování obrazu pomocí vlastní implementace OCL.

Kvůli relativní nepřesnosti, ale vysoké rychlosti, je tato implementace vhodná pro použití zejména při zpracování vstupů v reálném čase (kamerové vstupy, video záznamy, atp.). Detekované zájmové body nejsou natolik podrobné, aby mohly v této podobě s dostatečnou přesností analyzovat a rozpoznávat jednotlivé lidské obličeje mezi sebou. Nicméně mohou být užitečné například pro zmíněné mapování 3D modelů a zejména při rozpoznávání lidských emocí, jak je viditelné na Obr. 16 v kapitole 3.8.

5 Závěr

Rozborem metod v oblastech detekce obličejů a zpracování obrazu, z nichž použité jsou popsány v teoretické části této práce, se podařilo implementovat detektor význačných obličejových bodů pro mobilní zařízení platformy Android, který je dostatečně rychlý pro zpracování a detekci i v reálném čase. Implementace byla provedena pouze na základě studia a použití tříd knihovny OpenCV a využitím Android API. Rychlost detekce byla otestována na několika zařízeních při použití více přístupů k detekci a její přesnost byla testována porovnáním s výsledky referenční databáze společnosti BioID. Díky tomu, že je aplikace vytvořená v jazyce Java, je navíc velmi snadno přenositelná napříč platformami a pro porovnání byla testována i její desktopová verze.

Aplikace má své problémy, určitou mírou omezující přesnost detekce. Tyto, i s možným řešením, jsou popsány v předchozí kapitole. Vyplývají ze statického rozdělení obličeje a zásahu výrazných předmětu do detekovaných oblastí.

Metody použité pro vytvoření této práce jsou, jak základními metodami zpracování obrazu, tak založeny na strojovém učení. Právě tato dvě odvětví jsou spolu stále více spjata a otevírají nové možnosti. Bližším studiem algoritmů strojového učení, umělé inteligence a pokročilejších metod pro analýzu a zpracování obrazu, by bylo možné tuto práci dále rozvést a využít výsledných dat pro implementaci některého z praktických využití, zmíněných dříve.

Literatura

[1] GONZÁLEZ, Rafael C. , WOODS, Richard Eugene. *Digital Image Processing*. 3rd edition. Prentice Hall, 2007. ISBN 0-13-168728-X.

[2] TURK, Matthew, PENTLAND, Alex. *Eigenfaces for Recognition* [online]. Massachusetts Institute of Technology, 1991. doi:10.1162/jocn.1991.3.1.71 . Dostupné z: `http://www.face-rec.org/algorithms/PCA/jcn.pdf`

[3] VIOLA, P., JONES, M. Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition. *Proceedings of the 2001 IEEE Computer Society Conference*. on. vol. 1, 2001. pp. I-511-I-518 vol.1.

[4] ROTE, G. Computing the minimum Hausdorff distance between two point sets on a line under translation. In: *Information Processing Letters*. 1991, v. 38, pp. 123-127.

[5] JESORSKY, Oliver, KIRCHBERG, Klaus J., FRISCHHOLZ, Robert. (2001) Robust Face Detection Using the Hausforff Distance. In: *Third International Conference on Audio- and Video-based Biometric Person Authentication*. Halmstad, Sweden: Springer, 2001. pp. 90–95.

[6] FARKAS, Leslie G. *Anthropometry of the Head and Face*. New York: Raven Press, 1994. ISBN 9780781701594 .

[7] SUZUKI, Satoshi. (1983) Topological Structural Analysis of Digitized Binary Images by Border Following. In: *Computer vision, graphics, and image processing* pp. 30, 32-46. Japan: Shizuoka University, 2001.

[8] LÁNÍK, Aleš. *Extrakce obrazových příznaků* [online]. Brno: VUT, 2012. Dostupné z: `fit.vutbr.cz/study/courses/IKR/public/stare_prednasky_2012/04_obrazove_priznaky/ikr-obrazove-priznaky-2012.pdf`

[9] SCHAPIRE, Robert E. *Explaining AdaBoost* [online]. Princeton University, Department of Computer Science, 2015. Dostupné z: `http://rob.schapire.net/papers/explaining-adaboost.pdf`

[10] KOURKOUTIS, L. G., PANOULAS, K. I., HADJILEONTIA, L. Automated iris and gaze detection using chrominance: application to human-computer interaction using a low resolution webcam. In: *19 IEEE Int. Conf. on Tools with Artificial Intelligence*, 2007. pp 536-539 .

[11] *OpenCV: Face Detection using Haar Cascades* [online]. OpenCV 3.1.0 Documentation. Dostupné z: `http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html`.

[12] *OpenCV: Image Thresholding* [online]. OpenCV 3.1.0 Documentation. Dostupné z: http://docs.opencv.org/3.1.0/d7/d4d/tutorial_py_thresholding.html.

[13] *BioID Face Database* [online]. BioID AG, Switzerland. Dostupné z: https://www.bioid.com/About/BioID-Face-Database.

[14] *flandmark, Open-source implementation of facial landmarks detector* [online]. UŘIČÁK, Michal, FRANC, Vojtěch (Praha: ČVUT, 2015). Dostupné z: http://cmp.felk.cvut.cz/~uricamic/flandmark/.

[15] *CLandmark, Open Source Landmarking Library* [online]. UŘIČÁK, Michal, FRANC, Vojtěch (Praha: ČVUT, 2015). Dostupné z: http://cmp.felk.cvut.cz/~uricamic/clandmark/.

[16] *Facial Landmark Detection by Deep Multi-task Learning* [online]. Department of Information Engineering, The Chinese University of Hong Kong (2014). Dostupné z: http://mmlab.ie.cuhk.edu.hk/projects/TCDCN.html.

[17] *Active Shape Models with Stasm* [online]. MILBORROW, Stephen, NICOLLS, Fred. Dostupné z: http://www.milbo.users.sonic.net/stasm/.

[18] *Develop Apps, Android Developers* [online]. Google, Inc., Android API Documentation. Dostupné z: http://developer.android.com/develop/index.html.

[19] *FileOutputStream, Android Developers* [online]. Google, Inc., Android API Documentation. Dostupné z: http://developer.android.com/reference/java/io/FileOutputStream.html.

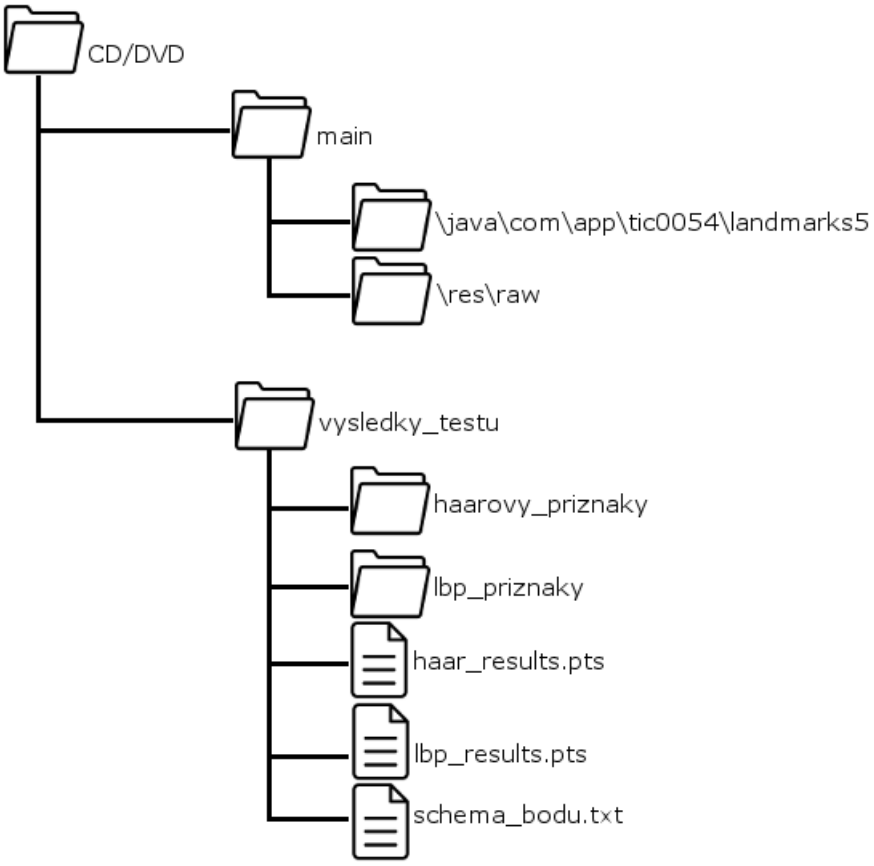
[20] *JavaCameraView* [online]. OpenCV 2.4.9 Android Documentation. Dostupné z: http://docs.opencv.org/java/2.4.9/org/opencv/android/JavaCameraView.html.

[21] *CameraBridgeViewBase.CvCameraViewListener2* [online]. OpenCV 3.0.0 Android Documentation. Dostupné z: http://docs.opencv.org/java/3.0.0/org/opencv/android/CameraBridgeViewBase.CvCameraViewListener2.html.

[22] *CascadeClassifier* [online]. OpenCV 2.4.11 Java Documentation. Dostupné z: http://docs.opencv.org/java/2.4.11/org/opencv/objdetect/CascadeClassifier.html.

[23] *Basic Structures* [online]. OpenCV 2.4.12 Documentation . Dostupné z: http://docs.opencv.org/2.4/modules/core/doc/basic_structures.html.

A Přílohy na CD/DVD



- Složka **main** obsahuje strukturu složky main z projektu Android Studia.

  Podsložka **landmarks5** obsahuje zdrojové kódy aplikace.

  Podsložka **raw** obsahuje XML soubory dat příznaků použitých pro detekci.

- Složka **vysledky_testu** obsahuje výsledky testů přesnosti detekce.

  Podsložka **haarovy_priznaky** obsahuje soubory formátu ".pts"obsahující souřadnice detekovaných zájmových bodů, při jejichž detekci bylo použito Haarových příznaků pro detekci obličeje.

  Podsložka **lbp_priznaky** obsahuje stejně jako výše, jen při použití LBP příznaků.

  Soubor **haar_results.pts** obsahuje výsledky porovnání souřadnic detekovaných bodů, při použití Haar. přízn., s referenční databází BioID.

  Soubor **lbp_results.pts** obsahuje stejně jako výše, jen při použití LBP přízn.

  Soubor **schema_bodu.txt** obsahuje popis významu jednotlivých řádku v souborech detekovaných bodů.